REMARKS

Status of the Claims

Claims 1-3, 5-10, 12-16, and 18-20 are currently present in the Application, and claims 1, 8, and 14 are independent claims. Claims 7, 13 and 20 have been amended to address an alleged lack of antecedent basis. No claims have been canceled or added in this Response. Applicants are not conceding in this Application that those claims are not patentable over the art cited by the Examiner, as the present claim amendments and cancellations are only for facilitating expeditious prosecution of the Application. Applicants respectfully reserve the right to pursue these and other claims in one or more continuation and/or divisional patent applications.

Claim Rejections - 35 U.S.C. § 112

Claims 7, 13, and 20 were rejected as allegedly having insufficient antecedent basis for the term "a third time." Applicants have amended the term "a third time" to read "a subsequent time." Applicants have also made it clear that this "subsequent time" occurs after the re-compilation limitation. As the limitation "a subsequent time" is introduced in claims 7, 13, and 20 and this term does not appear in intervening base claims of any of the three claim sets, Applicants submit that there is proper antecedent basis for the term "a subsequent time." Consequently, Applicants respectfully request that the Examiner withdraw the rejection under § 112 in light of Applicants' amendment.

Claim Rejections – Alleged Obviousness Under 35 U.S.C. § 103

Claims 1-3, 5-10, 12-16, and 18-20 were rejected under 35 U.S.C. § 103 as allegedly being obvious, and therefore unpatentable, over U.S. Patent Publ. No. 2002/0144240 to Lueh et al. (hereinafter "Lueh") in view of U.S. Patent Publ. No. 2004/0167945 to Alexander Garthwaite (hereinafter "Garthwaite"). Applicants respectfully traverse the rejections.

Applicants independent claims (claims 1, 8, and 14) are respectively directed to a method, information handling system, and computer program product that reclaims

memory occupied by Just-in-Time (JIT) compiled programs. Each of the independent claims sets forth substantially similar steps to reclaim the memory. Taking claim 1 as an exemplary claim, the limitations of which include:

- tracking a JIT compiled program, the tracking recording tracking data that includes a method name corresponding to the JIT compiled program and an address range that corresponds to the JIT compiled program;
- discarding one or more memory pages included in the address range;
- branching to an address included in one of the discarded pages, the branching resulting in a page fault;
- retrieving the method name corresponding to the address that resulted in the page fault;
- executing a method corresponding to the retrieved method name;
- memory mapping the JIT compiled program from a nonvolatile storage location to the address range using a special filesystem;
- prior to the discarding, receiving, at the special filesystem, an instruction to write (to nonvolatile storage) the one or more memory pages that are about to be discarded; and
- returning a response indicating successful completion of the instruction without writing any of the pages to the nonvolatile storage location.

The Office Action contends that Lueh teaches the limitation of "branching to an address included in one of the discarded pages, the branching resulting in a page fault" (citing paragraphs [0031] and [0032] of Lueh). However, an analysis of Lueh reveals that Lueh does not teach or suggest branching to an address in a discarded page which results in a page fault. Instead, Lueh teaches inserting program stubs that, when called,

cause the native code (e.g., a method that was called) to be re-compiled. Paragraphs [0031] and [0032] of Lueh read as follows:

> [0031] Referring now to FIG. 6, a JAVA object method invocation flow diagram according to one embodiment of the present invention is illustrated. As in the conventional method invocation flow diagram of FIG. 4, the illustrated invention embodiment includes an object 600 including one or more fields of data 603 as well as a reference or pointer 604 in a first object field to a method table 606. Method table 606 is constructed in the same manner as the method table 406 depicted in FIG. 4, containing an entry 608 for each method of the class located at an offset 609 within the table 606 facilitating the resolution of a location of a desired method's code 616. Similarly, prior to the first invocation of a method, the method's corresponding entry 608 in its class's method table 606 contains a symbolic reference to a stub 610 rather than direct reference to the method's executable native code 616.
>
> [0032] However, the illustrated embodiment contains additional program code inserted into stub 610 including a threshold check 611 to determine whether the total amount of space occupied by compiled native program code (native code space 614) exceeds a defined threshold, and a code space collection routine 612 to reclaim code space occupied by the native code of selected methods. In the illustrated embodiment, when a method is invoked for the first time during a virtual machine instance, a threshold check 611 is first performed and a method_collection routine 612 is conditionally invoked if the predefined native code space size threshold has been exceeded. Once the native code of any previously-invoked methods has been reclaimed, their corresponding method table entries are updated to re-reference compilation stub 610 just as when the method table 606 was first created by the JVM so that later invocations of the reclaimed method or methods will invoke the JIT compiler to re-compile their associated bytecode. Following the threshold check and conditional native code reclamation, a compilation routine 613 is performed which in turn invokes JVM dynamic compilation of the presently-invoked method. The method table entry 608 corresponding to the currently-invoked method may then be updated as described with reference to FIG. 4 to reference. In this manner the native code of selected methods is reclaimed, thus reducing the overall amount of native code stored within memory and consequently improving cache or other memory

performance and the efficiency with which retained native code is executed.

Paragraphs [0031] and [0032] is essentially Lueh's entire detailed description of Lueh's Fig. 6 which is reproduced below:

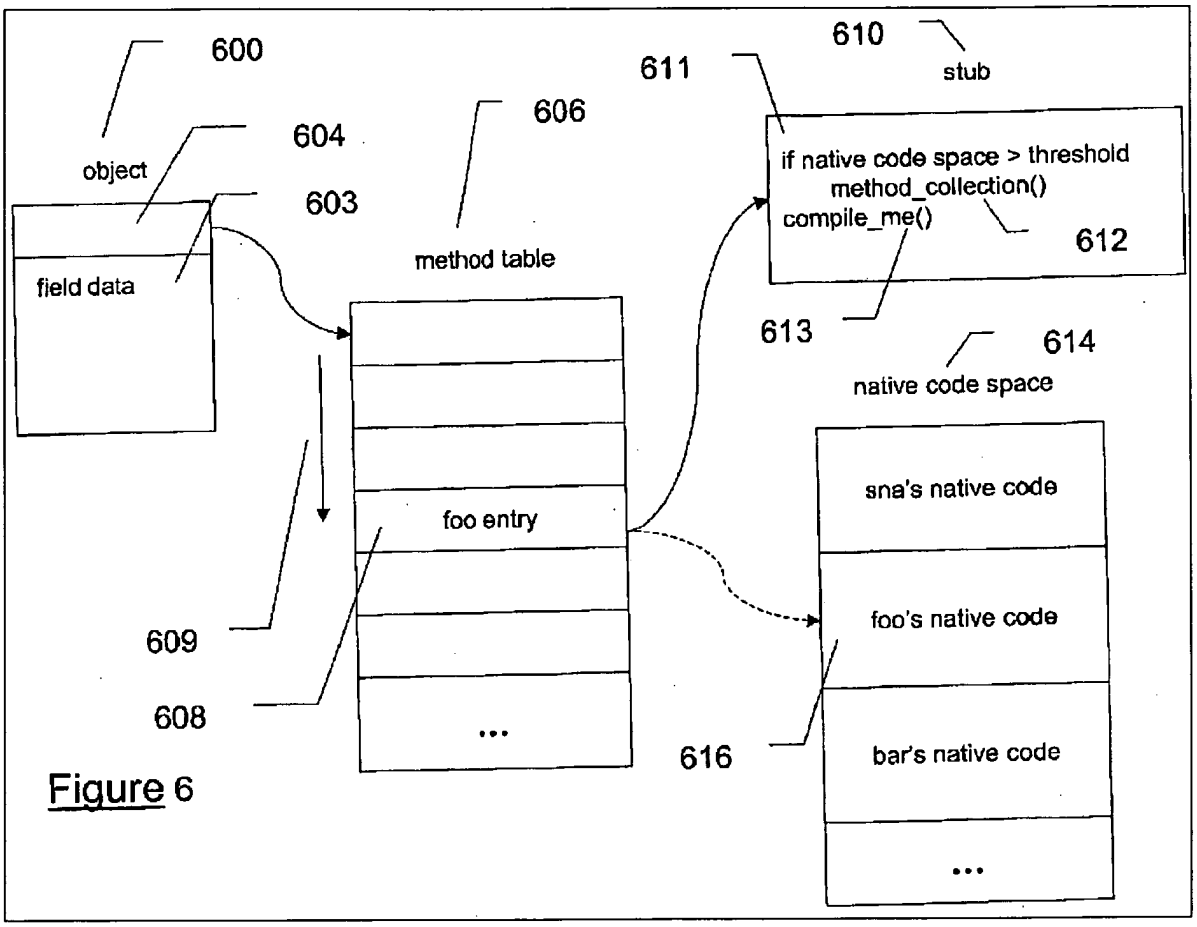

Figure 6

As can readily be seen, Leuh does not teach or suggest branching to an entry that results in a page fault. Instead, Leuh teaches a method table (606) with entries that point to program stub (stub 610) that cause compilation of the native code (native code 616). So, in the example presented by Leuh, if the "foo" entry is called, program stub 610 would be executed which would ***not*** result in a page fault and would instead result in foo's native code (616) being compiled. In essence, Leuh ***teaches away from***

Applicants invention because, rather than cause page faults, Leuh inserts program stubs, such as stub 610, which ***prevent*** page faults from occurring. Indeed, reviewing paragraphs [0031-0032] of the reference, Leuh never teaches or suggests that any page faults will result from any of the branching that occurs in Leuh's system.

As described above, Leuh simply does not teach or suggest the limitation of "branching to an address included in one of the discarded pages, the branching resulting in a page fault" as asserted in the Office Action. The Office Action does not contend that Garthwaite teaches or suggest this limitation and a review of Garthwaite reveals that Garthwaite does not teach or suggest this limitation. Therefore, because neither Leuh nor Garthwaite, alone or in combination with one another, teach or suggest Applicants' limitation of "branching to an address included in one of the discarded pages, the branching resulting in a page fault" which is included in each of Applicants' independent claims, Applicants respectfully submit that each of the independent claims is allowable over the combination of Lueh in view of Garthwaite.

Notwithstanding the reason for allowability noted above, Applicants further note that Garthwaite does not teach or suggest the limitation of "returning a response indicating successful completion of the instruction without writing any of the pages to the nonvolatile storage location," as contended in the Office Action (citing paragraph [0137] of Garthwaite).

Paragraph [0137] of Garthwaite reads as follows:

> [0137] Block 214 represents the operation of attempting to copy the referred-to collection-set object from the collection-set car to, say, FIG. 14's car 204. As was explained above, the referred-to object may have been evacuated already. If so, it will not be evacuated again: the evacuation will not succeed. Block 216 represents branching on whether the evacuation was successful. If it was not, a forwarding pointer will have been left in the already-evacuated object's previous location, and, as block 218 indicates, the collector will simply update the reference without duplicating the evacuation of the object to which it refers. If the evacuation was successful, the object will have been added to one of the LABs 206 into which the

car 204 has been divided, so the reference will be updated to reflect the new location, as block 219 indicates.

Paragraph [0137] is describing block 214 which is shown in Garthwaite's Fig. 15, which is reproduced below:

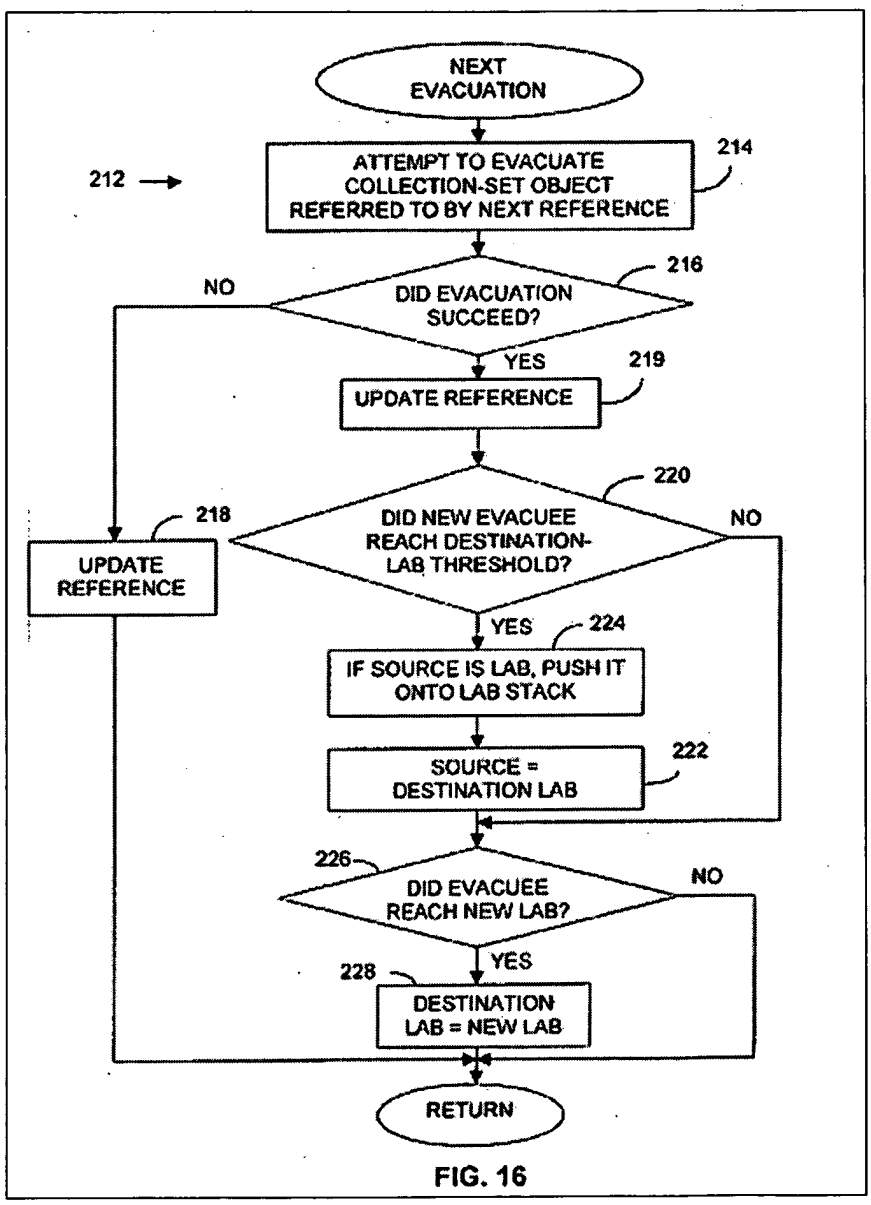

FIG. 16

The Office Action states that Garthwaite teaches Applicants' limitation citing Garthwaites teaching of "Block 216 represents branching on whether the evacuation was successful. If it was not, a forward pointer will have been left in the already-evacuated object's previous location, and, as block 218 indicates, the collector will simply update the reference without duplicating the object to which it refers." Garthwaite's Fig. 16 is teaching an evacuation process which is performed during garbage collection. The process shown in Fig. 16 is a routine that attempts to evacuate references in the Java garbage-collected heap. Block 216 is a decision block that determines whether a previous attempt to evacuate an object was successful (see block 214 of Fig. 16). If evacuation was not successful, the reference is updated (block 218), while if the evacuation was successful, the reference is updated (block 219) and further evacuation processing occurs. The cited section of Garthwaite is inapposite to Applicants' claimed limitation for several reasons. First, Applicants' limitation is returning a response after receiving "an instruction to write (to nonvolatile storage) the one or more memory pages that are about to be discarded," while Garthwaite is attempting to "evacuate" an object and does not teach or suggest handling an instruction to write memory pages that are about to be discarded. Second, Applicants' limitation returns "a response indicating successful completion of the instruction," while Garthwaite does not return any response indicating successful or unsuccessful completion of any instruction. Instead, Garthwaite is attempting to evacuate an object and is updating a reference if the evacuation does not succeed. Garthwaite is not responding to the completion of an instruction nor is Garthwaite responding to an instruction or command to write pages to a nonvolatile storage location.

The Office Action admits that Lueh does not teach or suggest Applicants' limitation of "returning a response indicating successful completion of the instruction without writing any of the pages to the nonvolatile storage location." As described above, Garthwaite also fails to teach or suggest this limitation. Therefore, in light of the fact that neither Lueh nor Garthwaite, alone or in combination with one another, teach or

suggest the aforementioned limitation which is included in each of Applicants' independent claims, Applicants respectfully submit that each of Applicants' independent claims are allowable over Lueh in view of Garthwaite.

As shown above, Applicants have overcome the rejections of the independent claims and have demonstrated that neither Lueh nor Garthwaite, alone or in combination with one another, teach or suggest all of the limitations set forth in Applicants' independent claims. Consequently, Applicants' independent claims are each allowable over the art of record. The remaining claims each depend, directly or indirectly, on one of the independent claims and, therefore, are each allowable for at least the same reasons that the independent claims are allowable.

Conclusion

As a result of the foregoing, it is asserted by Applicants that the remaining claims in the Application are in condition for allowance, and Applicants respectfully request an early allowance of such claims.

Applicants respectfully request that the Examiner contact the Applicants' attorney listed below if the Examiner believes that such a discussion would be helpful in resolving any remaining questions or issues related to this Application.

Respectfully submitted,

By /Joseph T. Van Leeuwen, Reg. No. 44,383/

Joseph T. Van Leeuwen, Reg. No. 44,383
Van Leeuwen & Van Leeuwen
Attorneys for Applicant
Telephone: (512) 301-6738
Facsimile: (512) 301-6742